

ANALYTICKÉ (A DALŠÍ POKROČILÉ) FUNKCE

Analytické vs. agregační funkce I.

- Podívejte se na výsledek těchto dvou SELECTů:

```
select nazev_pozice, count(*)  
from zamestnanec  
inner join zamestnani  
on zamestnani.zamestnani_id = zamestnanec.zamestnani_id  
group by nazev_pozice
```

```
select jmeno, prijmeni, nazev_pozice,  
       count(*) over (partition by nazev_pozice)  
from zamestnanec  
inner join zamestnani  
on zamestnani.zamestnani_id = zamestnanec.zamestnani_id;
```

Analytické vs. agregační funkce II.

- Při použití GROUP BY získáme agregované výsledky a celá matice je sgrupovaná podle sloupců uvedených v klauzuli GROUP BY, zatímco při použití analytické funkce jsme získali originální matici a agregované výsledky jsou její součástí.
- Vše je možné udělat i bez použití analytických funkcí, ale složitějším způsobem.
- Seznam všech analytických funkcí (na stránce úplně dole):
 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm

~~PARTITION BY I.~~

- Když nepoužijeme klauzuli PARTITION BY, pak se provede agregování bez sgrupování:

```
select jmeno, prijmeni, plat, nazev_pozice,  
avg(plat) over ()  
from zamestnanec  
inner join zamestnani  
on zamestnani.zamestnani_id = zamestnanec.zamestnani_id;
```

~~PARTITION BY~~ II.

- Když chcete s výsledkem analytické funkce dále pracovat, použijte vnořený select.
- Příklad: Získejte všechny zaměstnance, kteří mají nadprůměrný plat:

```
select * from (  
    select jmeno, prijmeni, plat, nazev_pozice,  
        round(avg(plat) over ()) avg_plat  
    from zamestnanec  
    inner join zamestnani  
    on zamestnani.zamestnani_id = zamestnanec.zamestnani_id  
) where plat > avg_plat  
order by plat desc;
```

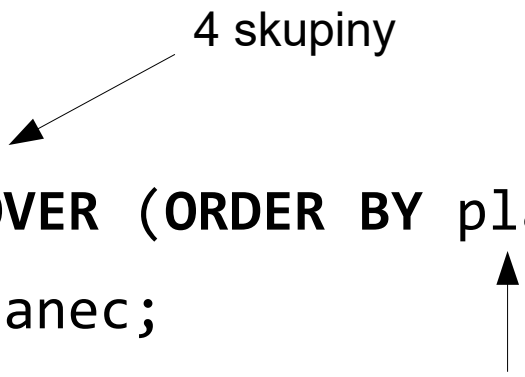
ORDER BY

- Některé analytické funkce vyžadují utřídění záznamů ve sgrupovaných skupinách. K tomu se používá klauzule ORDER BY uvnitř OVER (), která bude použita na následujících snímcích.

NTILE

- Analytická funkce NTILE rozděljuje vstupní data do definovaného počtu skupin, které jsou číslovány od jedničky.
- Příklad: Rozdělí zaměstnance do čtyř skupin podle velikosti jejich platu:

```
SELECT jmeno,  
       prijmeni,  
       plat,  
       ntile(4) OVER (ORDER BY plat DESC) skupina  
FROM zamestnanec;
```



Řazení do skupin bude podle velikosti platu sestupně (zaměstnanec s největším platem bude ve skupině č. 1, zaměstnanec s nejmenším platem bude ve skupině č. 4)

RANK

- Co když potřebujete přiřadit sekvenční pořadí (rank) lidem podle jejich zaměstnání na základě velikosti platu? K tomu slouží funkce RANK:

```
SELECT jmeno, prijmeni, plat, zamestnani.nazev_pozice,  
       rank() over  
       (partition BY zamestnanec.zamestnani_id order by plat)  
       AS poradi  
FROM zamestnanec  
INNER JOIN zamestnani  
ON zamestnani.zamestnani_id = zamestnanec.zamestnani_id;
```

- Všimněte si, že dva piloti mají stejný plat. Proto jim byl přiřazen stejný rank. Rank následujícího pilota ale nepokračuje plynule v číselné řadě.

DENSE_RANK

- Funkce DENSE_RANK funguje stejně jako RANK, ale přiřazuje po sobě jdoucí pořadí (rank), v sekvenci nejsou „díry“:

```
SELECT jmeno,  
       prijmeni,  
       plat,  
       zamestnani.nazev_pozice,  
       dense_rank() over  
         (partition BY zamestnanec.zamestnani_id order by plat)  
       AS poradi  
FROM zamestnanec  
INNER JOIN zamestnani  
ON zamestnani.zamestnani_id = zamestnanec.zamestnani_id;
```

ROW_NUMBER

- Co když potřebujete pouze vzestupnou sekvenci bez duplicitních čísel jako u rank i dense_rank?

```
SELECT jmeno,  
       prijmeni,  
       plat,  
       zamestnani.nazev_pozice,  
       row_number() over  
         (partition BY zamestnanec.zamestnani_id order by plat)  
       AS poradi  
FROM zamestnanec  
INNER JOIN zamestnani  
ON zamestnani.zamestnani_id = zamestnanec.zamestnani_id;
```

KEEP, FIRST, LAST

- KEEP funkce vrací první nebo poslední záznam utříděné skupiny. Používá se pro zjednodušení SELECTu při hledání minima nebo maxima (to samé by šlo self-joinem).
- Příklad: Zobrazí jména a příjmení všech zaměstnanců, jejich plat a minimální a maximální plat v pracovní pozici, kterou zaměstnanec zastává:

```
SELECT JMENO, PRIJMENI, ZAMESTNANI.NAZEV_POZICE, PLAT,  
MIN(PLAT) KEEP (DENSE_RANK FIRST ORDER BY PLAT) OVER (PARTITION BY  
NAZEV_POZICE) MIN_PLAT,  
MAX(PLAT) KEEP (DENSE_RANK LAST ORDER BY PLAT) OVER (PARTITION BY  
NAZEV_POZICE) MAX_PLAT  
FROM ZAMESTNANEC  
INNER JOIN ZAMESTNANI  
ON ZAMESTNANI.ZAMESTNANI_ID = ZAMESTNANEC.ZAMESTNANI_ID  
ORDER BY PLAT DESC
```

FIRST_VALUE

- Ke stejnému účelu můžete použít funkci FIRST_VALUE:

```
SELECT JMENO, PRIJMENI, ZAMESTNANI.NAZEV_POZICE, PLAT,  
FIRST_VALUE(PLAT) OVER (PARTITION BY NAZEV_POZICE ORDER  
BY PLAT ASC) MIN_PLAT,  
FIRST_VALUE(PLAT) OVER (PARTITION BY NAZEV_POZICE ORDER  
BY PLAT DESC) MAX_PLAT  
FROM ZAMESTNANEC  
INNER JOIN ZAMESTNANI  
ON ZAMESTNANI.ZAMESTNANI_ID = ZAMESTNANEC.ZAMESTNANI_ID  
ORDER BY PLAT DESC
```

LEAD, LAG

- Funkce LEAD a LAG naleznete v dokumentaci:
 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions074.htm
 - http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions070.htm#i1327527

RATIO_TO_REPORT

- Pomocí funkce RATIO_TO_REPORT jednoduše spočítáte procenta:

```
select last_name, salary, ratio_to_report(salary) over ()  
from hr.employees
```

- Vráťí platy všech zaměstnanců a hodnotu, která říká (v procentech) jak je velký plat každého zaměstnance oproti celkovému množství.

Jak zjistit duplicitní záznamy? I.

- Někdy se dostaneme do situace, kdy potřebujeme nejenom zjistit kolik je duplicitních záznamů (nebo je vyfiltrovat), ale také duplicitní záznamy fyzicky vypsát. Jak na to? Buď pomocí GROUP BY a vnořeného SELECTu (pomalejší způsob).
- Příklad: Kdy více zaměstnanců nastoupilo ve stejný den do zaměstnání?

```
SELECT *
```

```
FROM zamestnanec
```

```
WHERE datum_nastupu IN (SELECT datum_nastupu
```

```
FROM zamestnanec
```

```
GROUP BY datum_nastupu
```

```
HAVING COUNT(*) > 1)
```

Jak zjistit duplicitní záznamy? II.

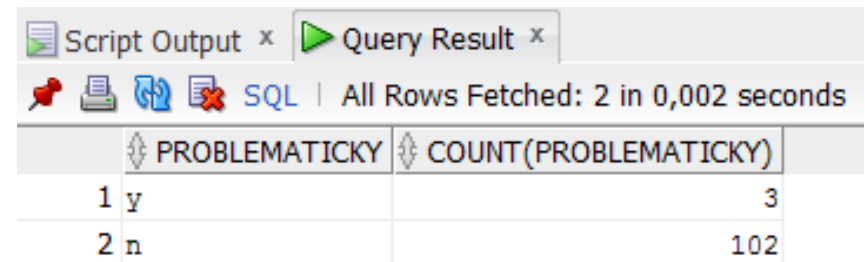
- Nebo efektivněji tímto způsobem:

```
SELECT *  
FROM (SELECT jmeno, prijmeni, datum_nastupu,  
count(*) over (partition by datum_nastupu) cnt  
FROM zamestnanec)  
WHERE cnt > 1;
```


PIVOT

- Pokud potřebujete v SELECTu provést transformaci řádků na sloupce, pak od Oracle 11g můžete použít PIVOT:

```
select problematicky,  
       count(problematicky)  
from pasazer  
group by problematicky;
```

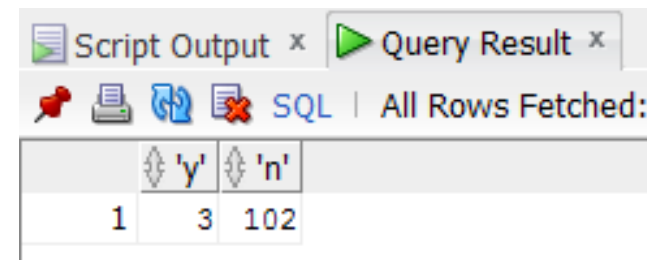


Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0,002 seconds

	PROBLEMATICKY	COUNT(PROBLEMATICKY)
1	y	3
2	n	102

```
select * from  
  (select problematicky  
   from pasazer)  
pivot (  
  count(problematicky)  
  for problematicky in ('y', 'n'));
```



Script Output x Query Result x

SQL | All Rows Fetched:

	'y'	'n'
1	3	102

UNPIVOT

- Samozřejmě také existuje funkce UNPIVOT, která provádí transformaci sloupců na řádky:
 - <http://www.oracle.com/technetwork/articles/sql/11g-pivot-097235.html>